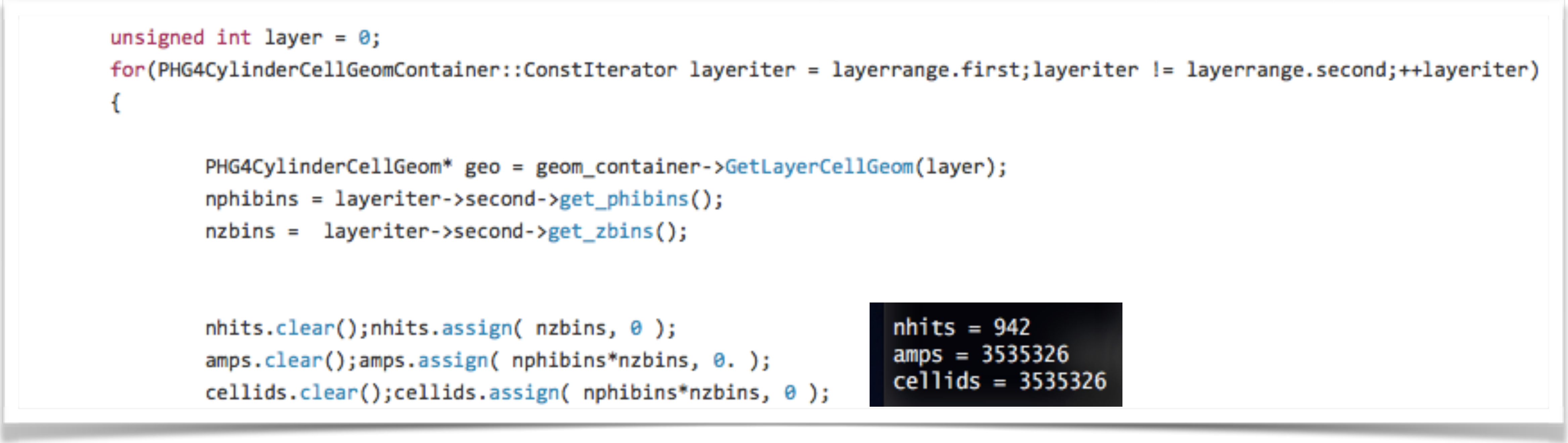
Previously Shown TPC Todos

(technical) fix the memory usage in central HIJING (actually from clusterizer?)
=> I will reassess after the meeting the clusterizer usage



```
unsigned int layer = 0;
for(PHG4CylinderCellGeomContainer::ConstIterator layeriter = layerrange.first;layeriter != layerrange.second;++layeriter)
{

        PHG4CylinderCellGeom* geo = geom_container->GetLayerCellGeom(layer);
        nphibins = layeriter->second->get_phibins();
        nzbins =  layeriter->second->get_zbins();


        nhits.clear();nhits.assign( nzbins, 0 );
        amps.clear();amps.assign( nphibins*nzbins, 0. );
        cellids.clear();cellids.assign( nphibins*nzbins, 0 );
```

```
nhits = 942
amps = 3535326
cellids = 3535326
```

(realism) add initial vertexing and remove perfect BBC input from tracking

(performance) improve the track fitting under occupancy, outlier rejection

(technical) improve the passing of uncertainties into HelixHough, remove hard coded errors in TPC version

(performance) remove vertex from fit

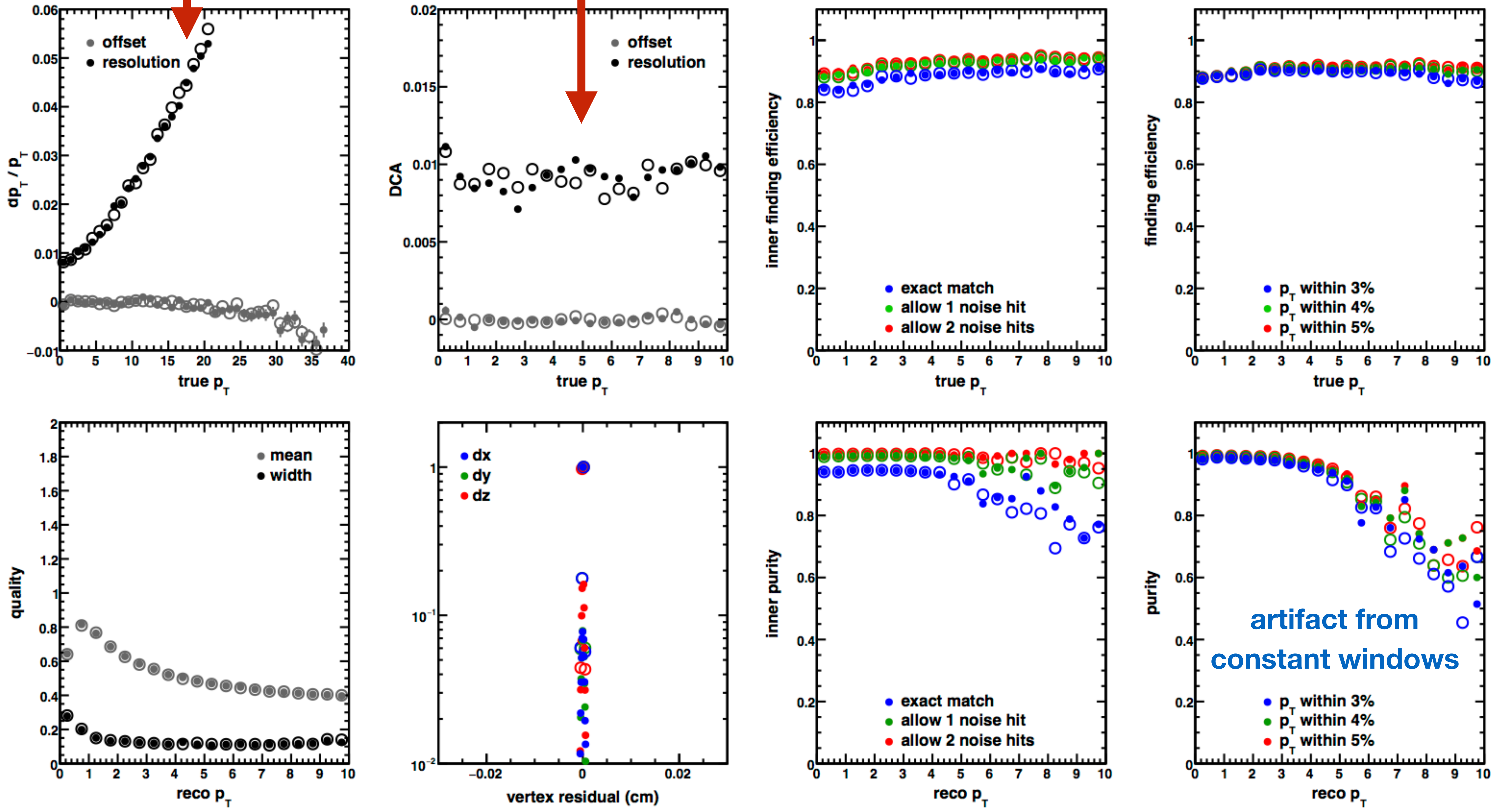
Mike: I want to keep pushing on pileup occupancy

Current TPC Performance

degraded momentum resolution under central event occupancy

fouled dca resolution under central event occupancy

“good” finding efficiencies

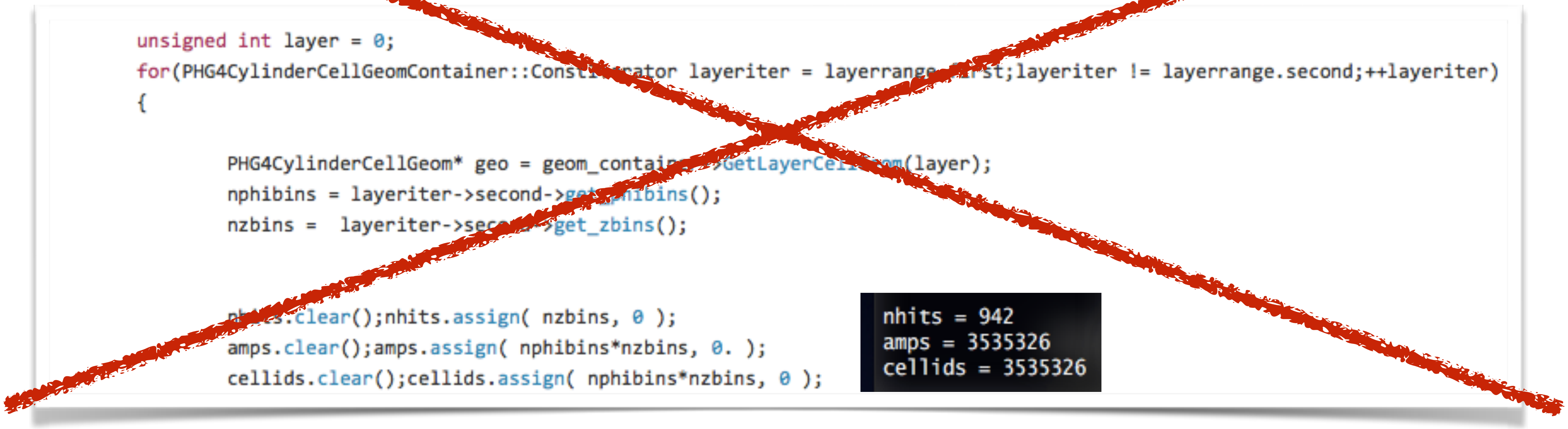


artifact from constant windows

ideal vertex resolution likely from perfect guessing

TPC Todos

~~(technical) fix the memory usage in central HIJING (actually from clusterizer?)~~
~~=> I will reassess after the meeting the clusterizer usage~~



```
unsigned int layer = 0;
for(PHG4CylinderCellGeomContainer::ConstIterator layeriter = layerrange.first;layeriter != layerrange.second;++layeriter)
{

        PHG4CylinderCellGeom* geo = geom_container->GetLayerCellGeom(layer);
        nphibins = layeriter->second->get_phibins();
        nzbins =  layeriter->second->get_zbins();


        nhits.clear();nhits.assign( nzbins, 0 );
        amps.clear();amps.assign( nphibins*nzbins, 0. );
        cellids.clear();cellids.assign( nphibins*nzbins, 0 );
```

```
nhits = 942
amps = 3535326
cellids = 3535326
```

(realism) add initial vertexing and remove perfect BBC input from tracking

(performance) improve the track fitting under occupancy, outlier rejection

(#2) (technical) improve the passing of uncertainties into HelixHough, remove hard coded errors in TPC version

(performance) remove vertex from fit

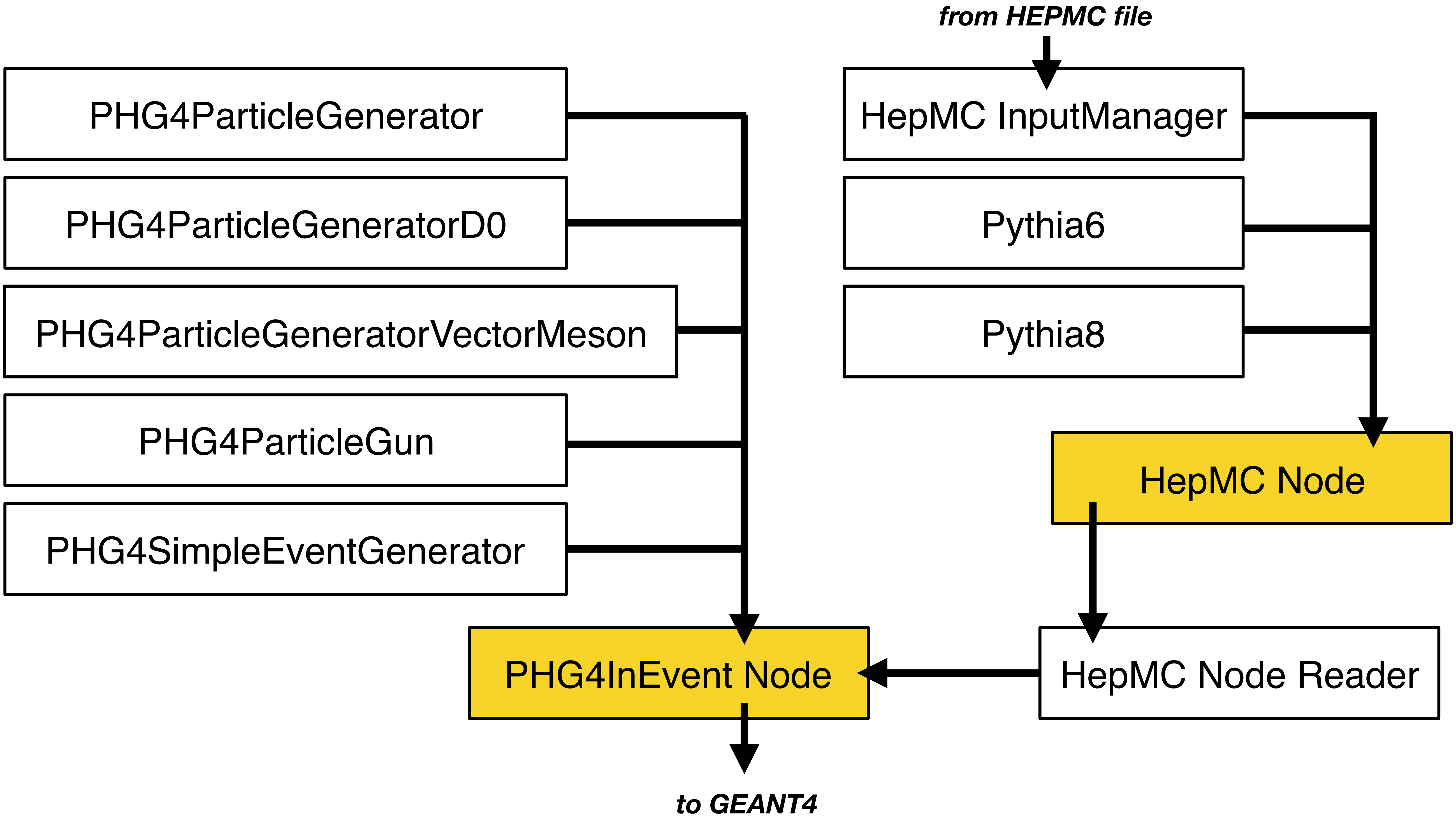
(#1) Mike: I want to keep pushing on pileup occupancy

Pileup Effort

(5) Pileup Simulations

(I) Add Time Dependence to g4main / g4detectors

(II) Revise Generator workflow

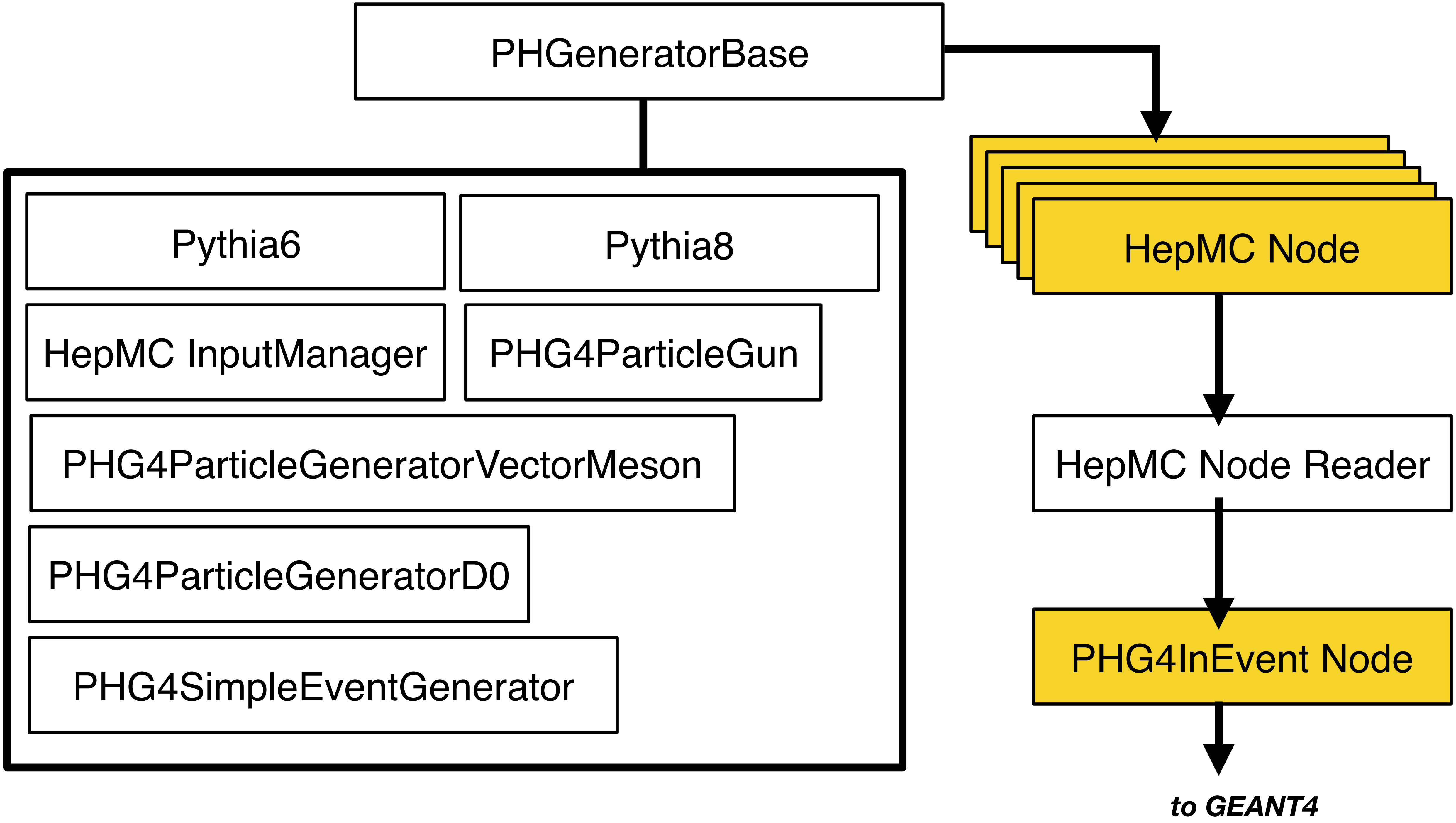


Pileup Effort II

(5) Pileup Simulations

(I) Add Time Dependence to g4main / g4detectors

(II) Revise Generator workflow

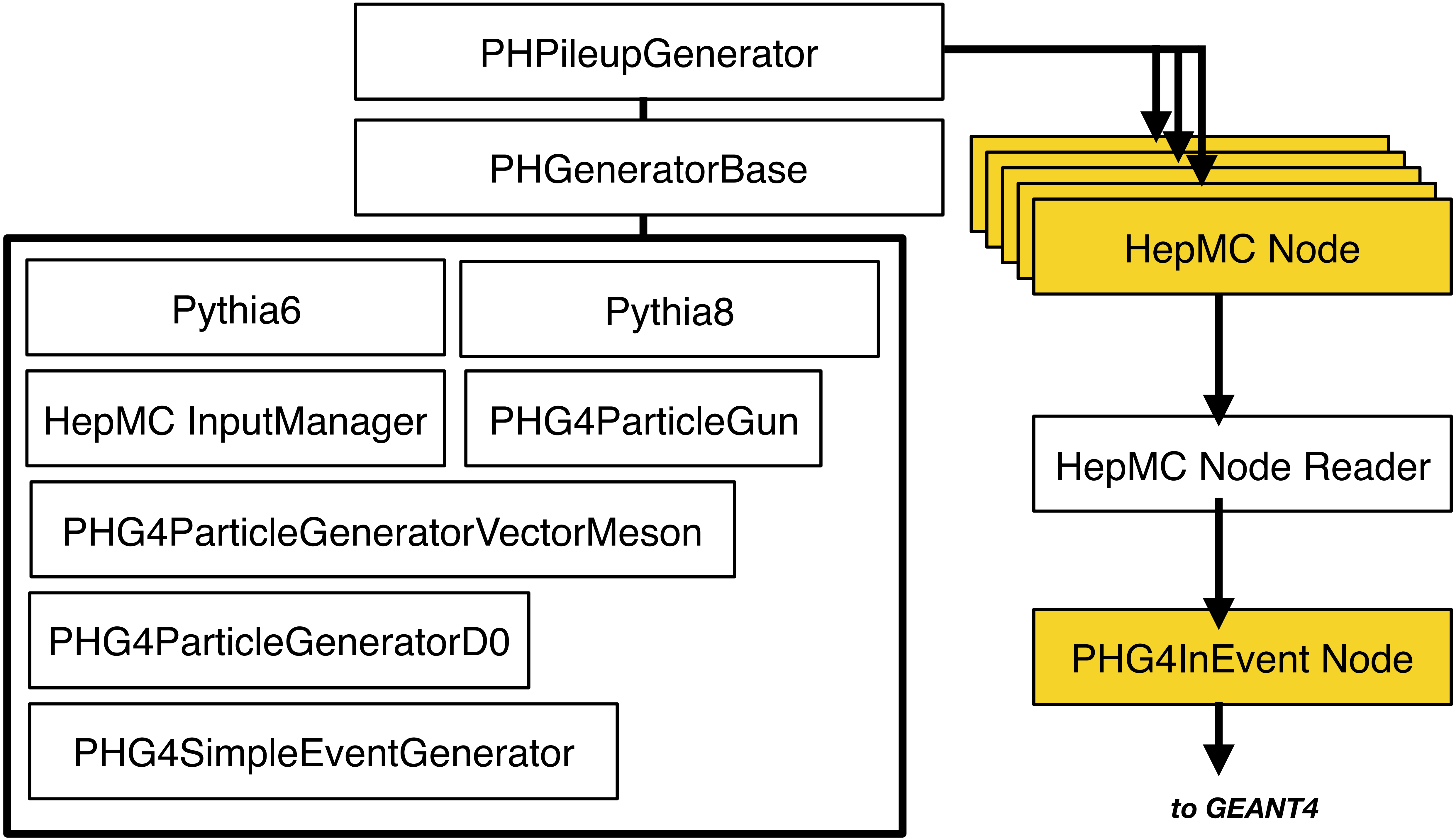


Pileup Effort III

(5) Pileup Simulations

(I) Add Time Dependence to g4main / g4detectors

(II) Revise Generator workflow



Pileup IV

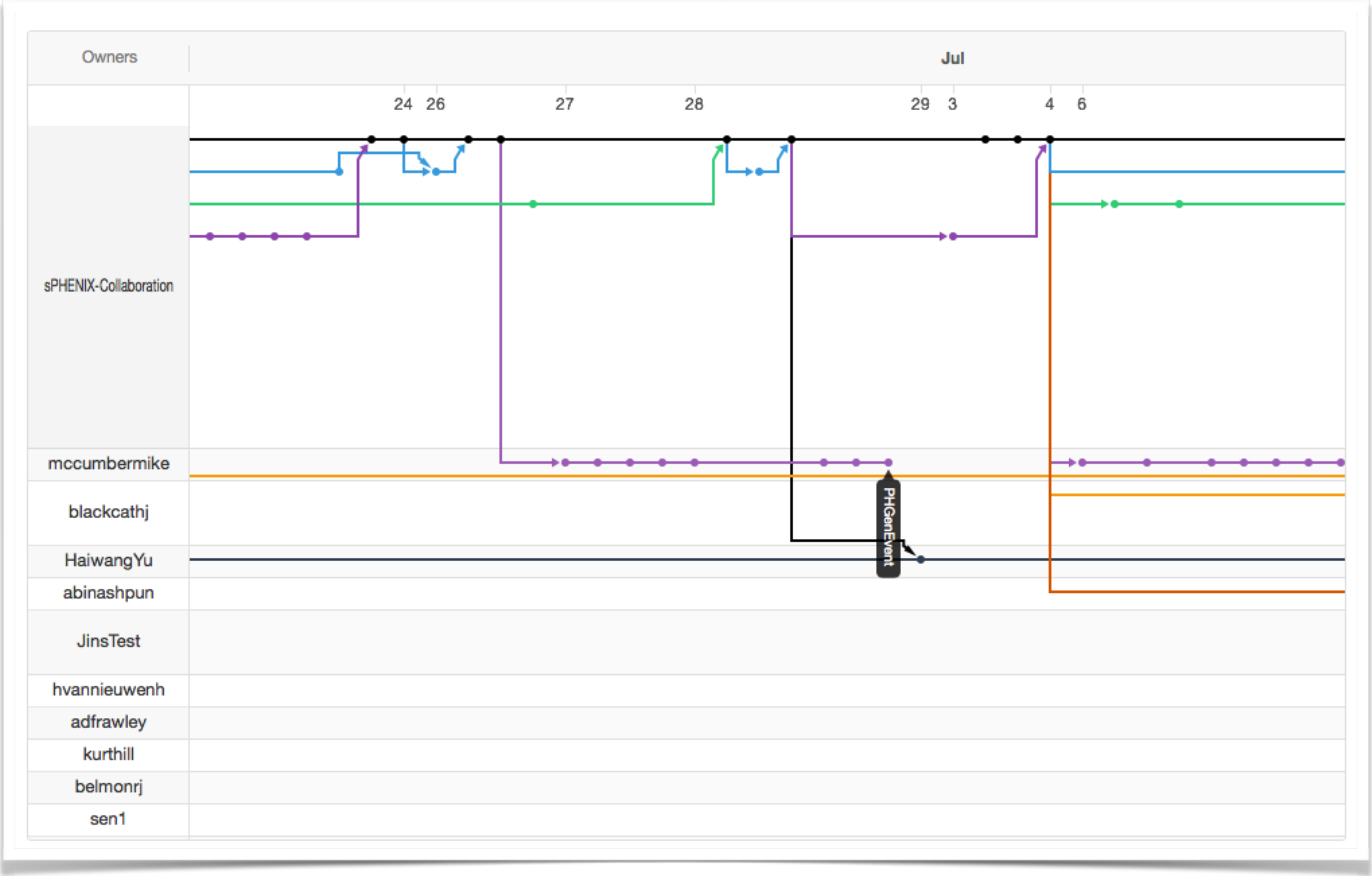
(5) Pileup Simulations

~~(I) Add Time Dependence to g4main / g4detectors~~

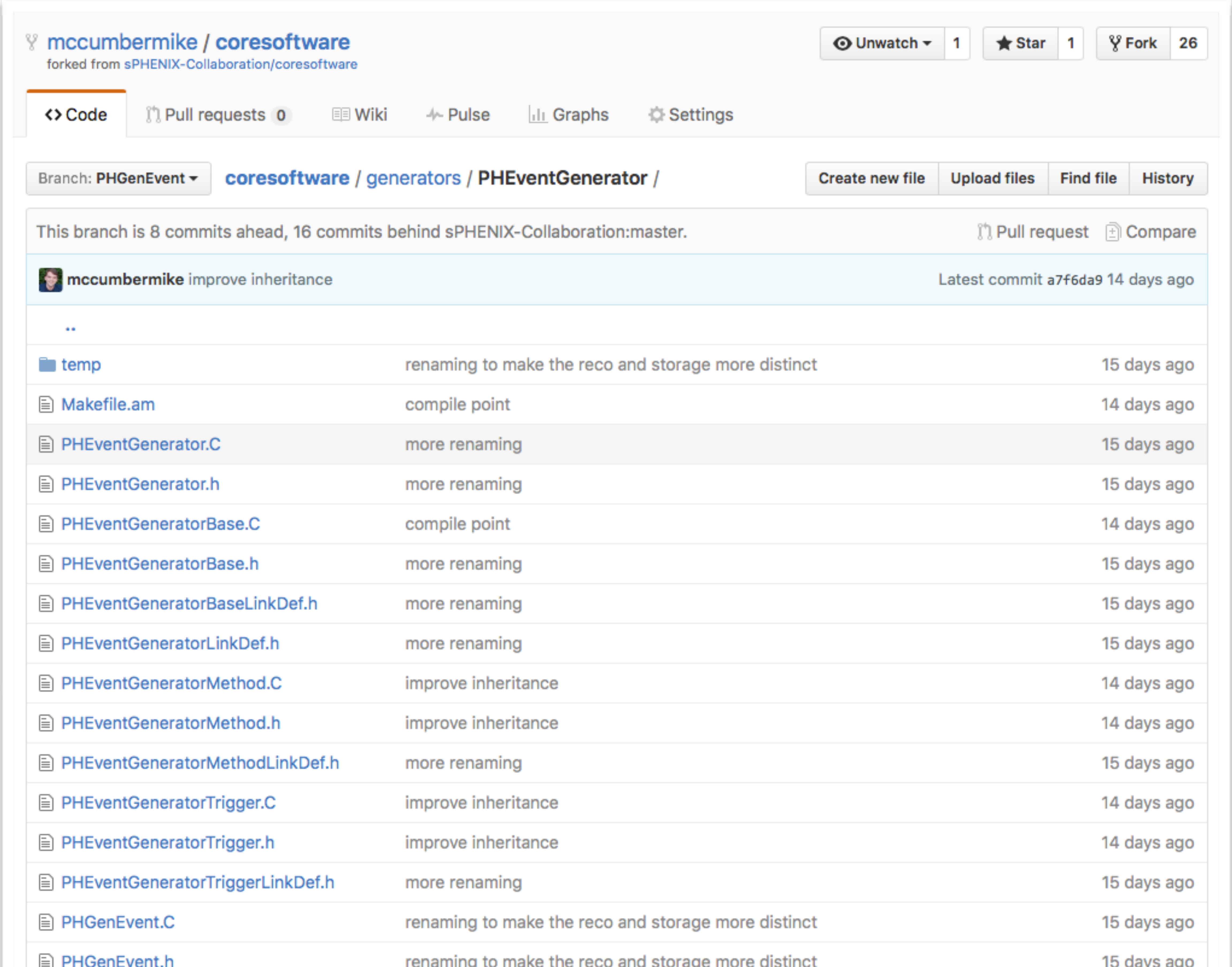
→ (II) Revise Generator workflow

(III) Requires Multiple Vertexing (**RAVE interface**)

Pileup Generator Branch



Pileup Generator Location



mccumbermike / coresoftware
forked from sPHENIX-Collaboration/coresoftware

coresoftware / generators / PHEventGenerator /

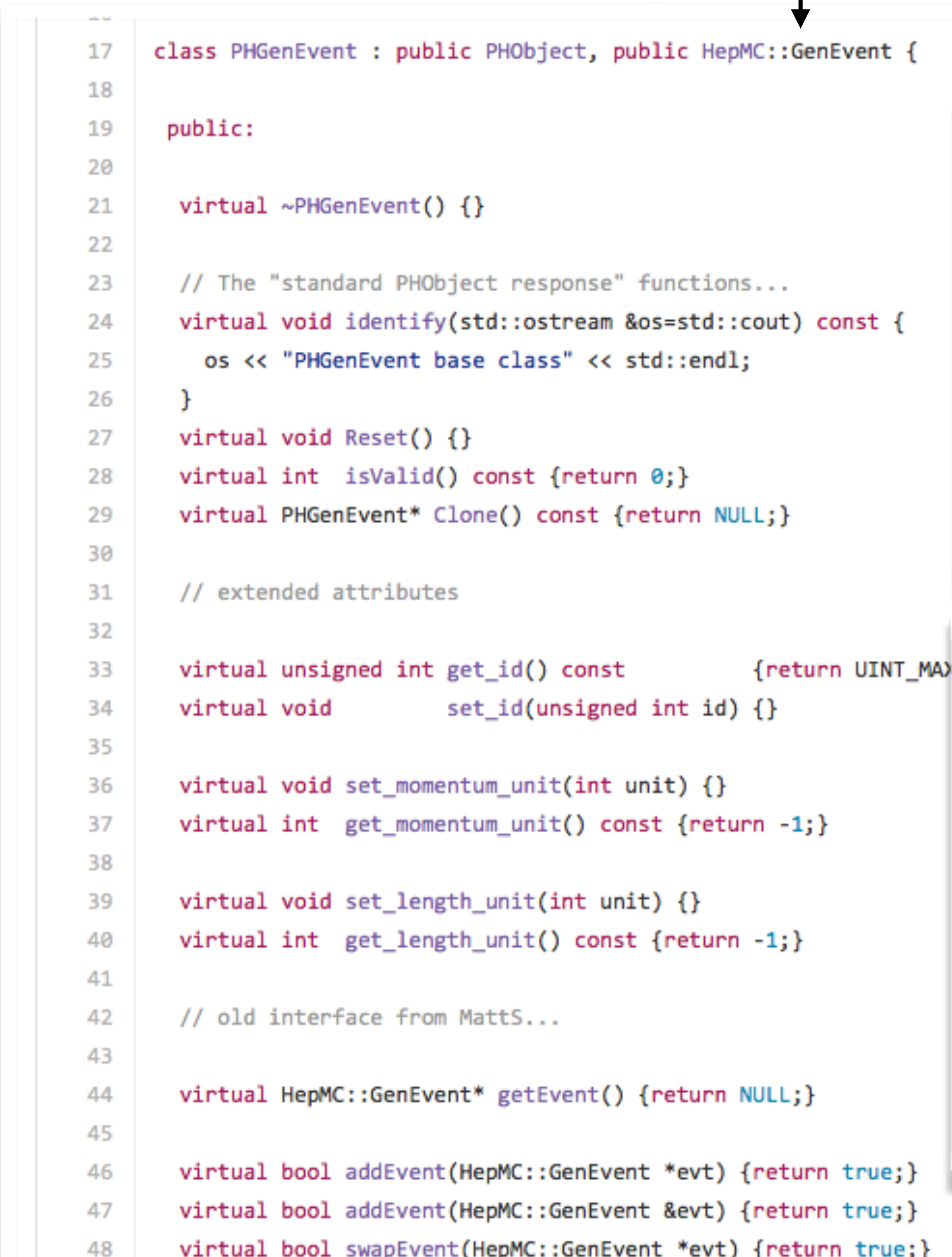
This branch is 8 commits ahead, 16 commits behind sPHENIX-Collaboration:master.

mccumbermike improve inheritance — Latest commit a7f6da9 14 days ago

..		
temp	renaming to make the reco and storage more distinct	15 days ago
Makefile.am	compile point	14 days ago
PHEventGenerator.C	more renaming	15 days ago
PHEventGenerator.h	more renaming	15 days ago
PHEventGeneratorBase.C	compile point	14 days ago
PHEventGeneratorBase.h	more renaming	15 days ago
PHEventGeneratorBaseLinkDef.h	more renaming	15 days ago
PHEventGeneratorLinkDef.h	more renaming	15 days ago
PHEventGeneratorMethod.C	improve inheritance	14 days ago
PHEventGeneratorMethod.h	improve inheritance	14 days ago
PHEventGeneratorMethodLinkDef.h	more renaming	15 days ago
PHEventGeneratorTrigger.C	improve inheritance	14 days ago
PHEventGeneratorTrigger.h	improve inheritance	14 days ago
PHEventGeneratorTriggerLinkDef.h	more renaming	15 days ago
PHGenEvent.C	renaming to make the reco and storage more distinct	15 days ago
PHGenEvent.h	renaming to make the reco and storage more distinct	15 days ago

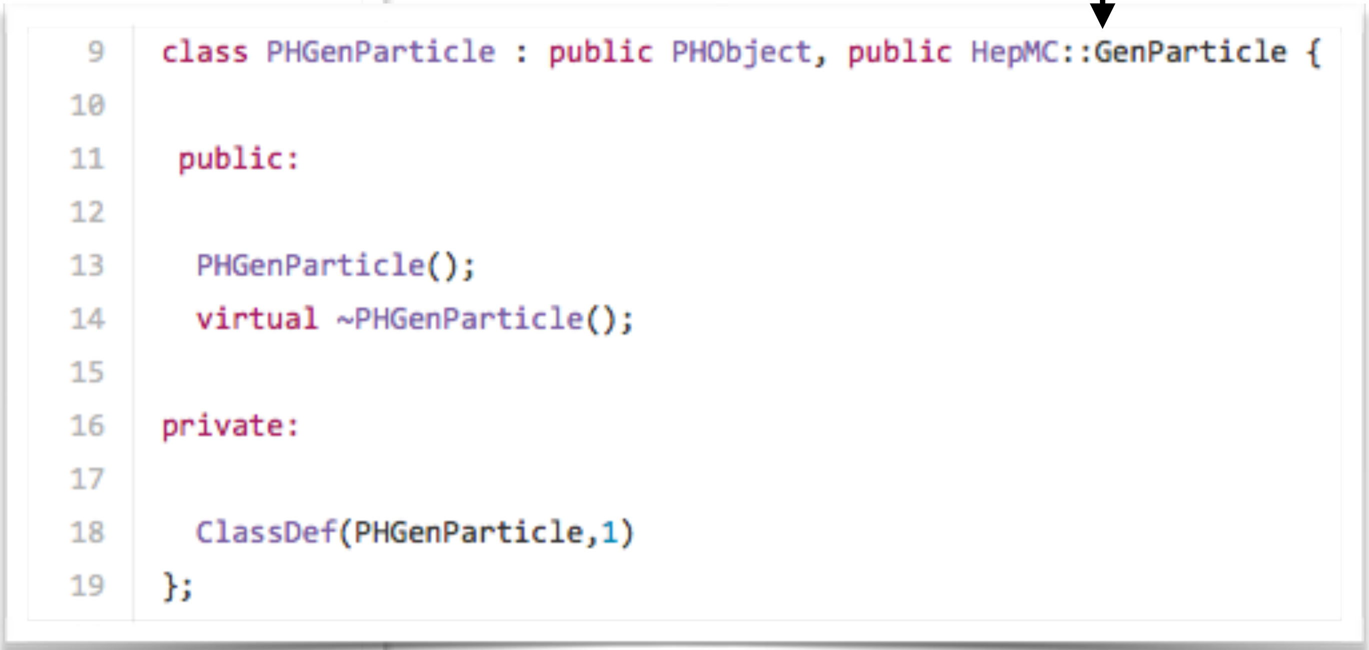
New Node Storage

Inherits from HepMC::GenEvent



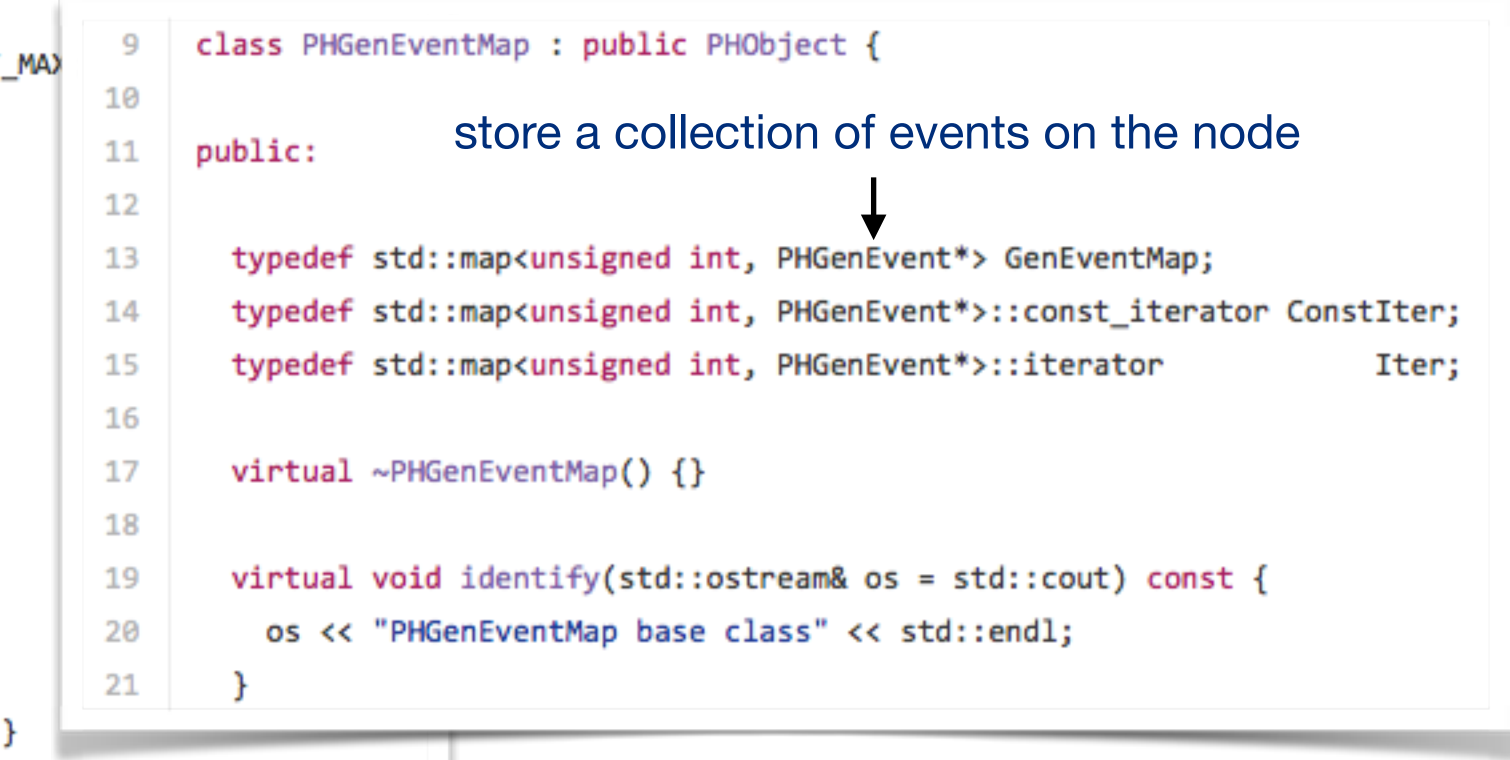
```
class PHGenEvent : public PHObject, public HepMC::GenEvent {

 public:

  virtual ~PHGenEvent() {}

  // The "standard PHObject response" functions...
  virtual void identify(std::ostream &os=std::cout) const {
    os << "PHGenEvent base class" << std::endl;
  }
  virtual void Reset() {}
  virtual int  isValid() const {return 0;}
  virtual PHGenEvent* Clone() const {return NULL;}

  // extended attributes

  virtual unsigned int get_id() const          {return UINT_MA
  virtual void         set_id(unsigned int id) {}

  virtual void set_momentum_unit(int unit) {}
  virtual int  get_momentum_unit() const {return -1;}

  virtual void set_length_unit(int unit) {}
  virtual int  get_length_unit() const {return -1;}

  // old interface from MattS...

  virtual HepMC::GenEvent* getEvent() {return NULL;}

  virtual bool addEvent(HepMC::GenEvent *evt) {return true;}
  virtual bool addEvent(HepMC::GenEvent &evt) {return true;}
  virtual bool swapEvent(HepMC::GenEvent *evt) {return true;}
```

Inherits from HepMC::GenParticle



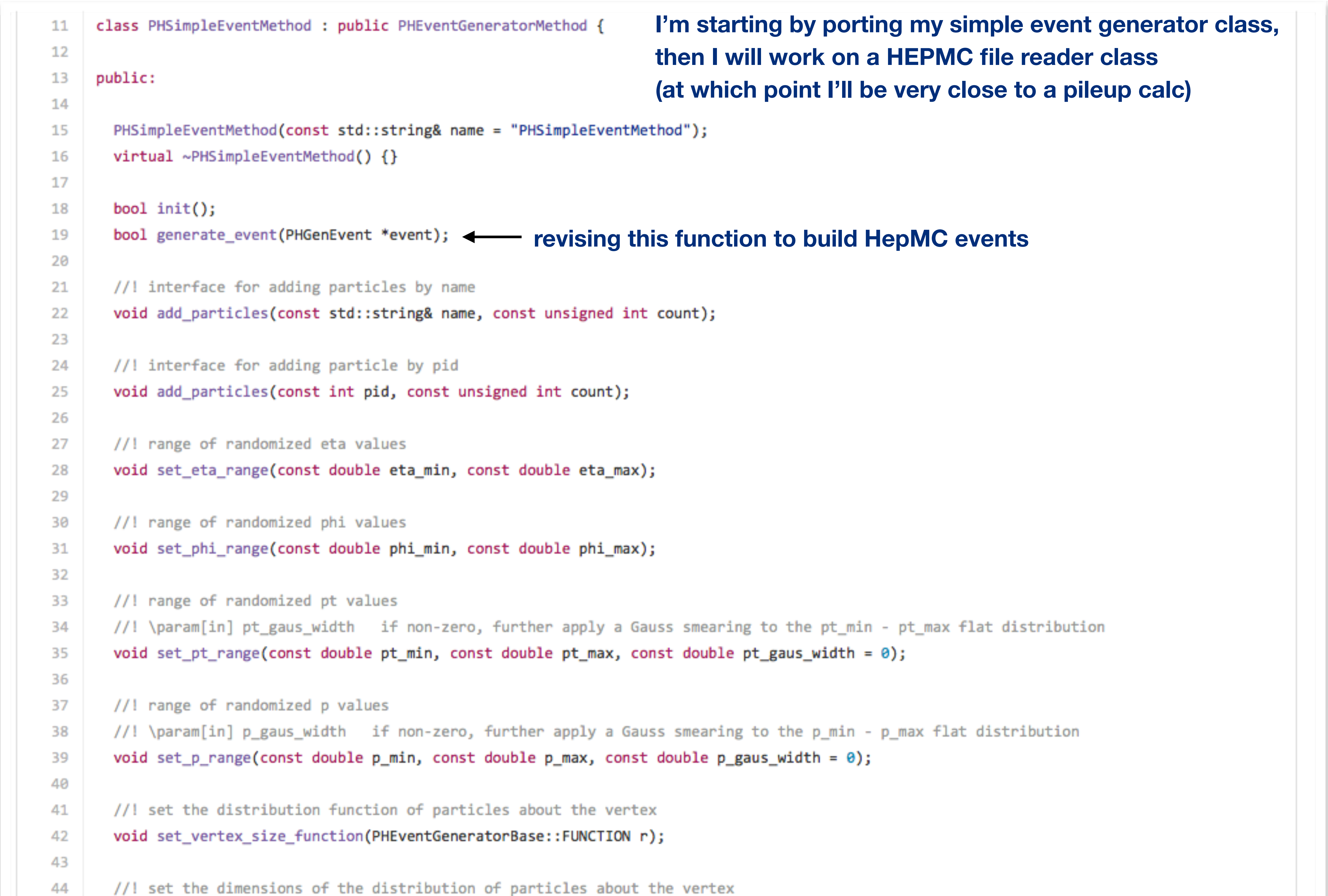
```
class PHGenParticle : public PHObject, public HepMC::GenParticle {

 public:

  PHGenParticle();
  virtual ~PHGenParticle();

private:

  ClassDef(PHGenParticle,1)
};
```

store a collection of events on the node



```
class PHGenEventMap : public PHObject {

public:

  typedef std::map<unsigned int, PHGenEvent*> GenEventMap;
  typedef std::map<unsigned int, PHGenEvent*>::const_iterator ConstIter;
  typedef std::map<unsigned int, PHGenEvent*>::iterator            Iter;

  virtual ~PHGenEventMap() {}

  virtual void identify(std::ostream& os = std::cout) const {
    os << "PHGenEventMap base class" << std::endl;
  }
```

First Method Class



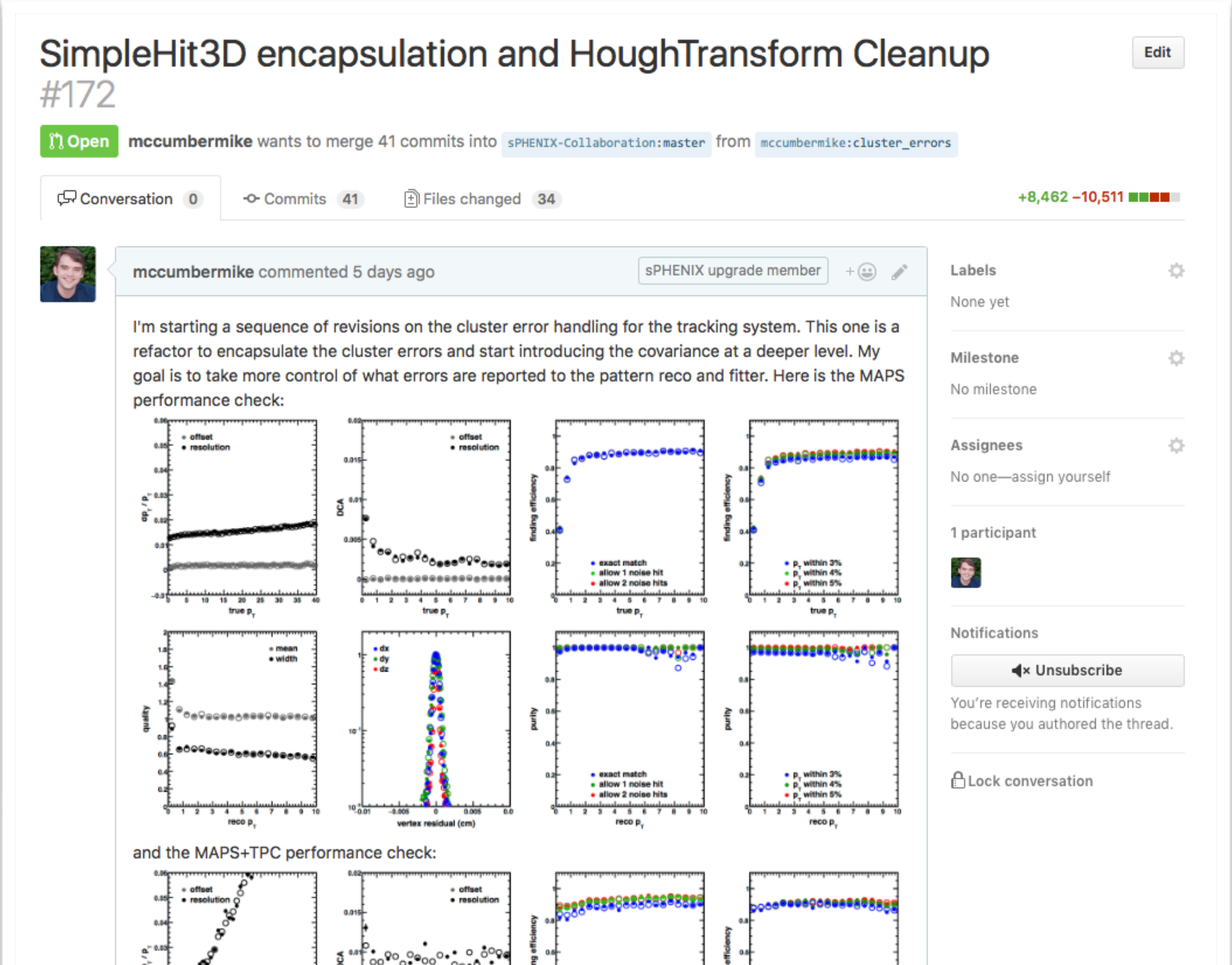
```
class PHSimpleEventMethod : public PHEventGeneratorMethod {

public:

  PHSimpleEventMethod(const std::string& name = "PHSimpleEventMethod");
  virtual ~PHSimpleEventMethod() {}

  bool init();
  bool generate_event(PHGenEvent *event);

  //! interface for adding particles by name
  void add_particles(const std::string& name, const unsigned int count);

  //! interface for adding particle by pid
  void add_particles(const int pid, const unsigned int count);

  //! range of randomized eta values
  void set_eta_range(const double eta_min, const double eta_max);

  //! range of randomized phi values
  void set_phi_range(const double phi_min, const double phi_max);

  //! range of randomized pt values
  //! \param[in] pt_gaus_width   if non-zero, further apply a Gauss smearing to the pt_min - pt_max flat distribution
  void set_pt_range(const double pt_min, const double pt_max, const double pt_gaus_width = 0);

  //! range of randomized p values
  //! \param[in] p_gaus_width   if non-zero, further apply a Gauss smearing to the p_min - p_max flat distribution
  void set_p_range(const double p_min, const double p_max, const double p_gaus_width = 0);

  //! set the distribution function of particles about the vertex
  void set_vertex_size_function(PHEventGeneratorBase::FUNCTION r);

  //! set the dimensions of the distribution of particles about the vertex
```

I’m starting by porting my simple event generator class, then I will work on a HEPMC file reader class (at which point I’ll be very close to a pileup calc)

← **revising this function to build HepMC events**

TPC Cluster Errors

a partial refactor of the cluster error passing will come in on #172
+ replaced ex,ey,ez errors with size covariance for full silicon tracker
+ next I will work on the TPC tracking



SimpleHit3D encapsulation and HoughTransform Cleanup #172

Edit

Open **mccumbermike** wants to merge 41 commits into `sPHENIX-Collaboration:master` from `mccumbermike:cluster_errors`

Conversation 0 | Commits 41 | Files changed 34 | +8,462 −10,511

mccumbermike commented 5 days ago — sPHENIX upgrade member

I'm starting a sequence of revisions on the cluster error handling for the tracking system. This one is a refactor to encapsulate the cluster errors and start introducing the covariance at a deeper level. My goal is to take more control of what errors are reported to the pattern reco and fitter. Here is the MAPS performance check:

and the MAPS+TPC performance check:

Labels
None yet

Milestone
No milestone

Assignees
No one—assign yourself

1 participant

Notifications
Unsubscribe
You're receiving notifications because you authored the thread.

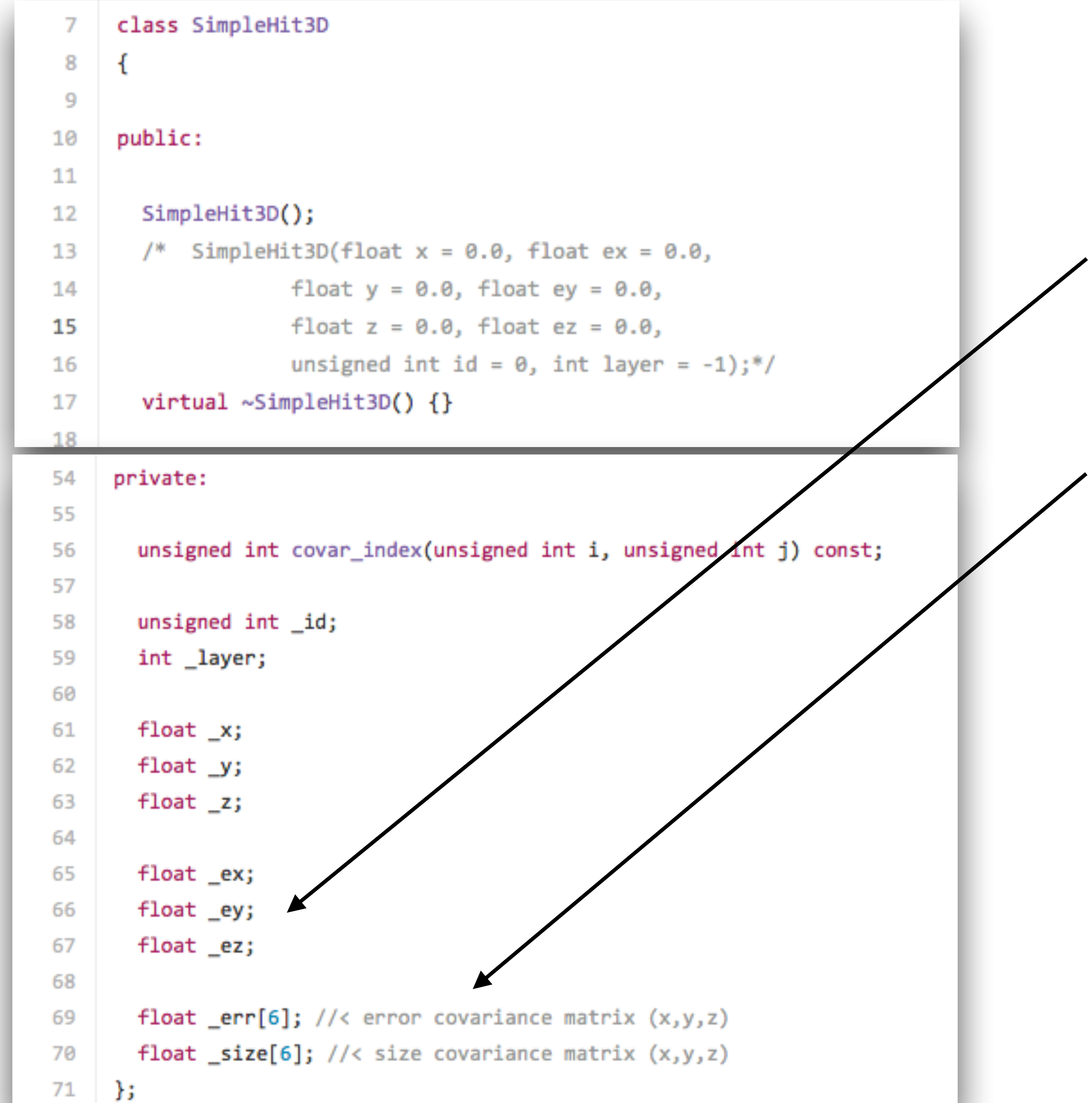
Lock conversation

Two Basic Issues

(1) Alan uses two senses of the uncertainty interchangeably in the code:

(i) pattern recognition cluster size

(ii) kalman fit position uncertainty



```
class SimpleHit3D
{

public:

  SimpleHit3D();
  /*  SimpleHit3D(float x = 0.0, float ex = 0.0,
              float y = 0.0, float ey = 0.0,
              float z = 0.0, float ez = 0.0,
              unsigned int id = 0, int layer = -1);*/
  virtual ~SimpleHit3D() {}

```

```
private:

  unsigned int covar_index(unsigned int i, unsigned int j) const;

  unsigned int _id;
  int _layer;

  float _x;
  float _y;
  float _z;

  float _ex;
  float _ey;
  float _ez;

  float _err[6]; //< error covariance matrix (x,y,z)
  float _size[6]; //< size covariance matrix (x,y,z)
};
```

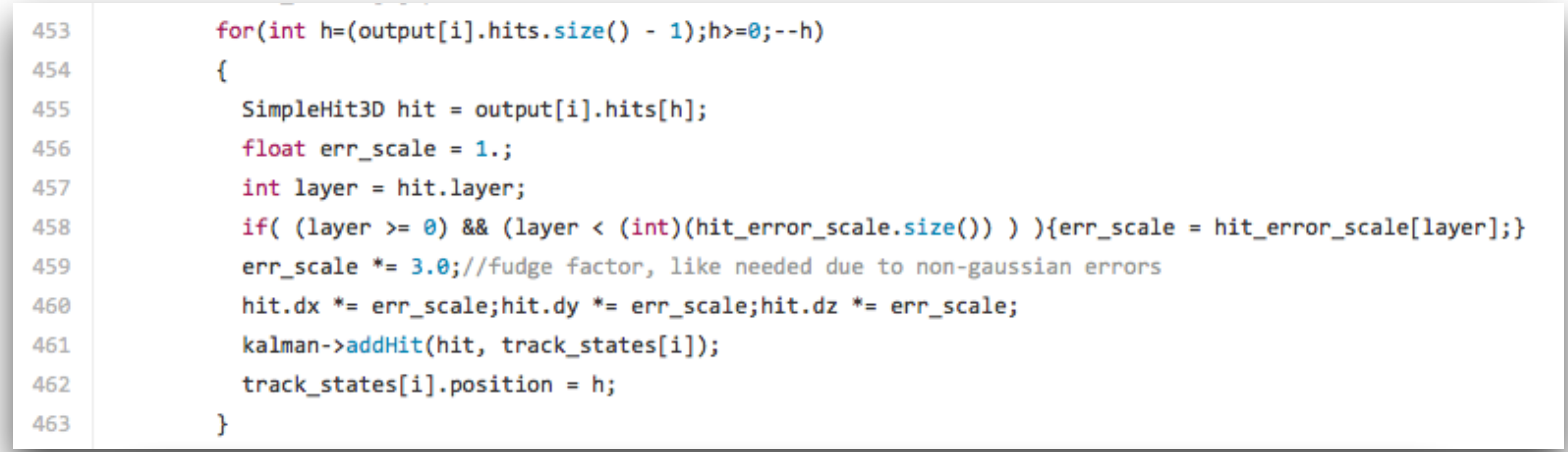
preserve Alan’s storage for now

add independent storage for the voting cluster size and the fitting position uncertainty

Two Basic Issues

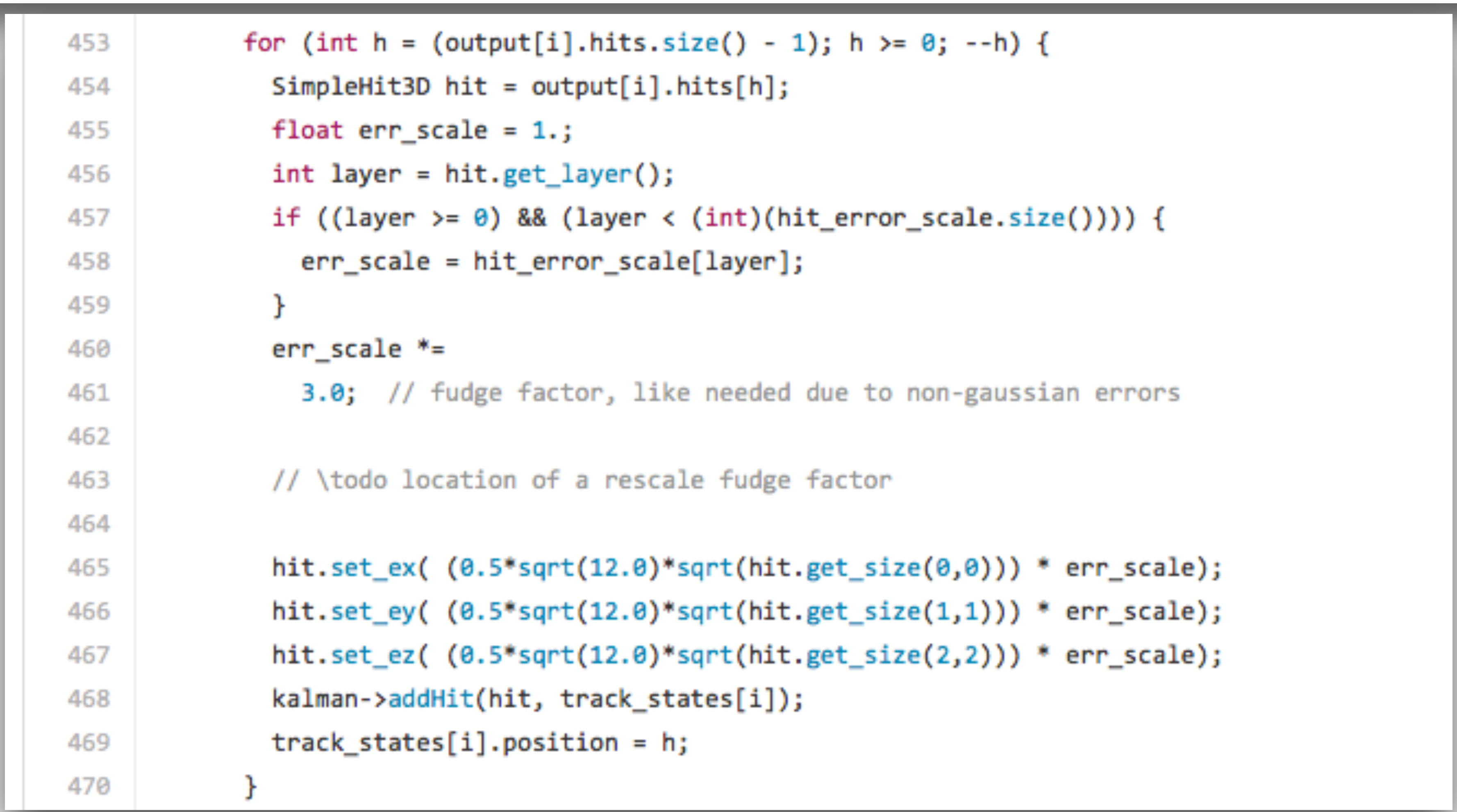
(2) Alan’s code scales arbitrarily between the two senses of the uncertainty as needed, but on the hit vector storage: very difficult to know at compile time what the stored uncertainties actually are!

Alan’s version



```
for(int h=(output[i].hits.size() - 1);h>=0;--h)
{
  SimpleHit3D hit = output[i].hits[h];
  float err_scale = 1.;
  int layer = hit.layer;
  if( (layer >= 0) && (layer < (int)(hit_error_scale.size()) ) ){err_scale = hit_error_scale[layer];}
  err_scale *= 3.0;//fudge factor, like needed due to non-gaussian errors
  hit.dx *= err_scale;hit.dy *= err_scale;hit.dz *= err_scale;
  kalman->addHit(hit, track_states[i]);
  track_states[i].position = h;
}
```

my version



```
for (int h = (output[i].hits.size() - 1); h >= 0; --h) {
  SimpleHit3D hit = output[i].hits[h];
  float err_scale = 1.;
  int layer = hit.get_layer();
  if ((layer >= 0) && (layer < (int)(hit_error_scale.size()))) {
    err_scale = hit_error_scale[layer];
  }
  err_scale *=
    3.0;  // fudge factor, like needed due to non-gaussian errors

  // \todo location of a rescale fudge factor

  hit.set_ex( (0.5*sqrt(12.0)*sqrt(hit.get_size(0,0))) * err_scale);
  hit.set_ey( (0.5*sqrt(12.0)*sqrt(hit.get_size(1,1))) * err_scale);
  hit.set_ez( (0.5*sqrt(12.0)*sqrt(hit.get_size(2,2))) * err_scale);
  kalman->addHit(hit, track_states[i]);
  track_states[i].position = h;
}
```

Next Steps

(1) I’ll finish off the cluster uncertainty passing in a few days, remove Alan’s error bars, remove the hard coded inputs to the PHG4HoughTransformTPC and create the covariances in the TPCClusterizer

At that point, the errors will be available throughout the processing with known constant values and refitting with Haiwang’s Kalman will be possible.

(2) After clearing this log jam for the TPC group, I’ll then return to my pileup generation effort and work again towards the pileup calculations we need for the MAPS + TPC tracker